RECEIVED
CENTRAL FAX CENTER

NOV 14 2005

Yee & Associates, P.C.

4100 Alpha Road
Suite 1100
Dallas, Texas 75244

Main No. (972) 385-8777
Facsimile (972) 385-7766

Facsimile Cover Sheet

To: Commissioner for Patents for **Examiner Lechi Truong** **Group Art Unit 2194**	Facsimile No.: **571/273-8300**
From: Carrie Parker Legal Assistant to Vicky Ash	No. of Pages Including Cover Sheet: **50**

Message:

Enclosed herewith:

- Transmittal Document; and
- Appeal Brief.

Re: Application No. **09/583,411**
Attorney Docket No: **AUS000153US1**

Date: Monday, November 14, 2005

Please contact us at (972) 385-8777 if you do not receive all pages indicated above or experience any difficulty in receiving this facsimile.

PLEASE CONFIRM RECEIPT OF THIS TRANSMISSION BY FAXING A CONFIRMATION TO 972-385-7766.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Taylor**

Serial No.: **09/583,411**

Filed: **May 31, 2000**

For: **Method and Apparatus for Bridging Service of Standard Object Identifier Based Protocols**

§ Group Art Unit: **2194**
§
§ Examiner: **Truong, Lechi**
§
§ Attorney Docket No.: **AUS000153US1**
§

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (571) 273-8300 on November 14, 2005.

By: Carrie Parker
Carrie Parker

TRANSMITTAL DOCUMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:
ENCLOSED HEREWITH:

- Appeal Brief (37 C.F.R. 41.37)

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account 09-0447.

Respectfully submitted,

Gerald H. Glanzman
Registration No. 25,035
Duke W. Yee
Registration No. 34,285
YEE & ASSOCIATES, P.C.
P.O. Box 802333
Dallas, Texas 75380
(972) 385-8777
ATTORNEY FOR APPLICANTS

Docket No. AUS000153US1

PATENT

RECEIVED
CENTRAL FAX CENTER
NOV 14 2005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Taylor**

Serial No. **09/583,411**

Filed: **May 31, 2000**

For: **Method and Apparatus for Bridging Service of Standard Object Identifier Based Protocols**

Group Art Unit: **2194**

Examiner: **Truong, Lechi**

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (571) 273-8300 on November 14, 2005.

By: Carrie Parker
Carrie Parker

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on September 12, 2005.

The fees required under § 41.20(B)(2), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-57

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: NONE
2. Claims withdrawn from consideration but not canceled: NONE
3. Claims pending: 1-57
4. Claims allowed: NONE
5. Claims rejected: 1-57
6. Claims objected to: NONE

C. CLAIMS ON APPEAL

The claims on appeal are: 1-57

STATUS OF AMENDMENTS

No amendments were made after the Final Office Action dated June 13, 2005.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to a method on a server (**104, 114, 200**) in a distributed data processing system (**100**) for maintaining a logical composite repository of Object Identifier (OID) tree structures (**Figure 3A and 3B**) (see *Specification*, page 8, 1-18 and page 13, lines 9-19). The method comprises receiving, in an OID abstraction layer (**414**), an OID tree structure from a repository (**408, 410, 412**). The OID abstraction layer is capable of receiving queries for objects in two or more different protocols (**402, 404, 406**) and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) (**416**) requests that the repository understands (see *Specification*, page 13, line 1, through page 16, line 25). The OID tree structure is registered with a registry associated with the OID abstraction layer (see *Specification*, page 15, lines 24-31 and page 16, lines 8-10) and the OID tree structure is added to a repository associated with the OID abstraction layer (see *Specification*, page 4, lines 10-16).

B. CLAIM 20 - INDEPENDENT

The subject matter of claim 20 is directed to an apparatus on a server (**104, 114, 200**) in a distributed data processing system (**100**) for maintaining a logical composite repository of Object Identifier (OID) tree structures (**Figure 3A and 3B**) (see *Specification*, page 8, 1-18 and page 13, lines 9-19). The apparatus comprises an OID abstraction layer (**414**) that receives an OID tree structure from a repository (**408, 410, 412**). The OID abstraction layer is capable of receiving queries for objects in two or more different protocols (**402, 404, 406**) and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) (**416**) requests that the repository understands (see *Specification*, page 13, line 1, through page 16, line 25). The apparatus comprises a registry associated with the OID abstraction layer that registers the OID tree structure (see *Specification*, page 15, lines 24-31 and page 16, lines 8-10) and provides a means for adding the OID tree structure to a repository associated with the OID abstraction layer (see *Specification*, page 4, lines 10-16).

C. CLAIM 39 - INDEPENDENT

The subject matter of claim 39 is directed to a computer program product in a computer readable medium for maintaining a logical composite repository of Object Identifier (OID) tree structures (**Figure 3A** and **3B**) (see *Specification*, page 8, 1-18 and page 13, lines 9-19). The computer program product provides instructions for receiving, in an OID abstraction layer (**414**), an OID tree structure from a repository (**408, 410, 412**). The OID abstraction layer is capable of receiving queries for objects in two or more different protocols (**402, 404, 406**) and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) (**416**) requests that the repository understands (see *Specification*, page 13, line 1, through page 16, line 25). The computer program product provides instructions for registering the OID tree structure with a registry associated with the OID abstraction layer (see *Specification*, page 15, lines 24-31 and page 16, lines 8-10) and the computer program product provides instructions for adding the OID tree structure to a repository associated with the OID abstraction layer (see *Specification*, page 4, lines 10-16).

D. CLAIM 9 - INDEPENDENT

The subject matter of claim 9 is directed to a method on a server (**104, 114, 200**) in a distributed data processing system (**100**) for retrieving object data from a repository (**408, 410, 412**). The method comprises receiving a first query for the object data from a requester in the distributed data processing system (see *Specification*, page 19, lines 3-5). The first query is in a protocol (**402, 404, 406**) recognized by an OID abstraction layer (**414**) (see *Specification*, page 14, line 31, through page 15, line 15). The OID abstraction layer is capable of receiving queries for objects in two or more different protocols (**402, 404, 406**) and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) (**416**) requests that the repository understands (see *Specification*, page 13, line 1 through page 16, line 25 and page 19, lines 8-10). The method comprises interpreting the first query according to the protocol recognized by the OID abstraction layer (see *Specification*, page 15, lines 18-23). The protocol recognized by the OID abstraction layer is one of the two or more different protocols (see *Specification*, page 8, lines 7-11). The method comprises locating a

repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer (see *Specification*, page 15, lines 24-31; page 16, lines 8-10; page 17, lines 21-23; and page 19, lines 5-9). The method comprises retrieving the object data from the repository using an OID abstraction layer application program interface (API) (see *Specification*, page 16, lines 14-25 and page 19, lines 8-18).

E. CLAIM 28 - INDEPENDENT

The subject matter of claim 28 is directed to an apparatus on a server (**104, 114, 200**) in a distributed data processing system (**100**) for retrieving object data from a repository (**408, 410, 412**). The apparatus provides a means for receiving a first query for the object data from a requester in the distributed data processing system (see *Specification*, page 19, lines 3-5). The first query is in a protocol (**402, 404, 406**) recognized by an OID abstraction layer (**414**) (see *Specification*, page 14, line 31, through page 15, line 15). The OID abstraction layer is capable of receiving queries for objects in two or more different protocols (**402, 404, 406**) and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) (**416**) requests that the repository understands (see *Specification*, page 13, line 1 through page 16, line 25 and page 19, lines 8-10). The apparatus provides a means for interpreting the first query according to the protocol recognized by the OID abstraction layer (see *Specification*, page 15, lines 18-23). The protocol recognized by the OID abstraction layer is one of the two or more different protocols (see *Specification*, page 8, lines 7-11). The apparatus provides a means for locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer (see *Specification*, page 15, lines 24-31; page 16, lines 8-10; page 17, lines 21-23; and page 19, lines 5-9). The apparatus provides a means for retrieving the object data from the repository using an OID abstraction layer application program interface (API) (see *Specification*, page 16, lines 14-25 and page 19, lines 8-18).

F. CLAIM 47 - INDEPENDENT

The subject matter of claim 47 is directed to a computer program product in a computer readable medium for retrieving object data from a repository (**408, 410, 412**). The computer program

product provides instructions for receiving a first query for the object data from a requester in the distributed data processing system (see *Specification*, page 19, lines 3-5). The first query is in a protocol (**402, 404, 406**) recognized by an OID abstraction layer (**414**) (see *Specification*, page 14, line 31, through page 15, line 15). The OID abstraction layer is capable of receiving queries for objects in two or more different protocols (**402, 404, 406**) and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) (**416**) requests that the repository understands (see *Specification*, page 13, line 1 through page 16, line 25 and page 19, lines 8-10). The computer program product provides instructions for interpreting the first query according to the protocol recognized by the OID abstraction layer (see *Specification*, page 15, lines 18-23). The protocol recognized by the OID abstraction layer is one of the two or more different protocols (see *Specification*, page 8, lines 7-11). The computer program product provides instructions for locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer (see *Specification*, page 15, lines 24-31; page 16, lines 8-10; page 17, lines 21-23; and page 19, lines 5-9). The computer program product provides instructions for retrieving the object data from the repository using an OID abstraction layer application program interface (API) (see *Specification*, page 16, lines 14-25 and page 19, lines 8-18).

G. CLAIM 6 - DEPENDENT

The subject matter of claim 6, which depends from claim 1 through dependent claim 5, is directed to a method wherein the OID abstraction layer (**414**) receives the information retrieved from the repository through the API (**416**) and encapsulates the information in a reply message to a target protocol interface, wherein the reply message is formatted for an appropriate protocol (**402, 404, 406**) for the target protocol interface, and wherein the appropriate protocol is one of the two or more different protocols (see *Specification*, page 10, lines 20-27 and page 19, lines 12-18).

H. CLAIM 7 - DEPENDENT

The subject matter of claim 7, which depends from claim 1, is directed to a method wherein the OID abstraction layer (**414**) receives a request for object data from a requesting protocol

interface, interprets the request according to a protocol (**402, 404, 406**) of the requesting protocol interface, wherein the protocol of the requesting protocol interface is one of the two or more different protocols, converts the request into an application program interface (API) (**414**) request that is forwarded to the repository (**408, 410, 412**), and receives an API reply from the repository having the object data (see *Specification*, page 8, lines 12-18 and page 15, lines 18-31).

I. CLAIM 8 - DEPENDENT

The subject matter of claim 8, which depends from claim 1 through dependent claim 7, is directed to a method wherein the OID abstraction layer (**414**) reformats the object data in a reply message according to the protocol (**402, 404, 406**) of the requesting protocol interface and sends the reply message to the requesting protocol interface (see *Specification*, page 16, lines 14-25 and page 19, lines 12-18).

J. CLAIM 14 - DEPENDENT

The subject matter of claim 14, which depends from claim 9 through dependent claims 10, 12, and 13, is directed to a method wherein the first reply is transformed into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer (**414**), and wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols (**402, 404, 406**) (see *Specification*, page 16, lines 14-25 and page 19, lines 12-18).

K. CLAIM 16 - DEPENDENT

The subject matter of claim 16, which depends from claim 9, is directed to a method wherein each repository in a plurality of repositories (**408, 410, 412**) contains information representing an Object Identifier (OID) subtree structure (**Figures 3A and 3B**), and wherein the plurality of repositories are formatted to support the two or more different protocols (**402, 404, 406**) (see *Specification*, page 14, lines 14-18 and page 15, lines 7-17).

L. CLAIM 19 - DEPENDENT

The subject matter of claim 19, which depends from claim 9, is directed to a method wherein Common Information Model used in conjunction with eXtendable Markup Language (CIM/XML) (406) is a protocol recognized by the OID abstraction layer (414) (see *Specification*, page 15, lines 18-23).

M. CLAIM 25 - DEPENDENT

The subject matter of claim 25, which depends from claim 20 through dependent claim 24, is directed to an apparatus wherein the OID abstraction layer (414) receives the information retrieved from the repository through the API (416) and encapsulates the information in a reply message to a target protocol interface, wherein the reply message is formatted for an appropriate protocol (402, 404, 406) for the target protocol interface, and wherein the appropriate protocol is one of the two or more different protocols (see *Specification*, page 10, lines 20-27 and page 19, lines 12-18).

N. CLAIM 26 - DEPENDENT

The subject matter of claim 26, which depends from claim 20, is directed to an apparatus wherein the OID abstraction layer (414) receives a request for object data from a requesting protocol interface, interprets the request according to a protocol (402, 404, 406) of the requesting protocol interface, wherein the protocol of the requesting protocol interface is one of the two or more different protocols, converts the request into an application program interface (API) (414) request that is forwarded to the repository (408, 410, 412), and receives an API reply from the repository having the object data (see *Specification*, page 8, lines 12-18 and page 15, lines 18-31).

O. CLAIM 27 - DEPENDENT

The subject matter of claim 27, which depends from claim 20 through dependent claim 26, is directed to an apparatus wherein the OID abstraction layer (414) reformats the object data in a

reply message according to the protocol (402, 404, 406) of the requesting protocol interface and sends the reply message to the requesting protocol interface (see *Specification*, page 16, lines 14-25 and page 19, lines 12-18).

P. CLAIM 33 - DEPENDENT

The subject matter of claim 33, which depends from claim 28 through dependent claims 29, 31, and 32, is directed to an apparatus wherein the first reply is transformed into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer (414), and wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols (402, 404, 406) (see *Specification*, page 16, lines 14-25 and page 19, lines 12-18).

Q. CLAIM 35 - DEPENDENT

The subject matter of claim 35, which depends from claim 28, is directed to an apparatus wherein each repository in a plurality of repositories (408, 410, 412) contains information representing an Object Identifier (OID) subtree structure (**Figures 3A** and **3B**), and wherein the plurality of repositories are formatted to support the two or more different protocols (402, 404, 406) (see *Specification*, page 14, lines 14-18 and page 15, lines 7-17).

R. CLAIM 38 - DEPENDENT

The subject matter of claim 38, which depends from claim 28, is directed to an apparatus wherein Common Information Model used in conjunction with eXtendable Markup Language (CIM/XML) (406) is a protocol recognized by the OID abstraction layer (414) (see *Specification*, page 15, lines 18-23).

S. CLAIM 44 - DEPENDENT

The subject matter of claim 44, which depends from claim 39 through dependent claim 43, is directed to a computer program product wherein the OID abstraction layer (414) receives the information retrieved from the repository through the API (416) and encapsulates the information

in a reply message to a target protocol interface, wherein the reply message is formatted for an appropriate protocol (402, 404, 406) for the target protocol interface, and wherein the appropriate protocol is one of the two or more different protocols (see *Specification*, page 10, lines 20-27 and page 19, lines 12-18).

T. CLAIM 45 - DEPENDENT

The subject matter of claim 7, which depends from claim 39, is directed to a computer program product wherein the OID abstraction layer (414) receives a request for object data from a requesting protocol interface, interprets the request according to a protocol (402, 404, 406) of the requesting protocol interface, wherein the protocol of the requesting protocol interface is one of the two or more different protocols, converts the request into an application program interface (API) (414) request that is forwarded to the repository (408, 410, 412), and receives an API reply from the repository having the object data (see *Specification*, page 8, lines 12-18 and page 15, lines 18-31).

U. CLAIM 46 - DEPENDENT

The subject matter of claim 8, which depends from claim 39 through dependent claim 45, is directed to a computer program product wherein the OID abstraction layer (414) reformats the object data in a reply message according to the protocol (402, 404, 406) of the requesting protocol interface and sends the reply message to the requesting protocol interface (see *Specification*, page 16, lines 14-25 and page 19, lines 12-18).

V. CLAIM 52 - DEPENDENT

The subject matter of claim 52, which depends from claim 47 through dependent claims 48, 50, and 51, is directed to a computer program product wherein the first reply is transformed into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer (414), and wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols (402, 404, 406) (see *Specification*, page 16, lines 14-25 and page 19, lines 12-18).

W. CLAIM 54 - DEPENDENT

The subject matter of claim 54, which depends from claim 47, is directed to a computer program product wherein each repository in a plurality of repositories (**408, 410, 412**) contains information representing an Object Identifier (OID) subtree structure (**Figures 3A** and **3B**), and wherein the plurality of repositories are formatted to support the two or more different protocols (**402, 404, 406**) (see *Specification*, page 14, lines 14-18 and page 15, lines 7-17).

X. CLAIM 57 - DEPENDENT

The subject matter of claim 57, which depends from claim 47, is directed to a computer program product wherein Common Information Model used in conjunction with eXtendable Markup Language (CIM/XML) (**406**) is a protocol recognized by the OID abstraction layer (**414**) (see *Specification*, page 15, lines 18-23).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. GROUND OF REJECTION 1 (Claims 1, 9, 20, and 39)

The Final Office Action rejects claims 1, 9, 20, and 39 under 35 U.S.C. 103(a) as being allegedly unpatentable over *Spofford* et al. (U.S. Patent 5,913,037) in view of *Dobbins* et al. (U.S. Patent 5,951,649) and further in view of *Pearson* (U.S. Patent 6,023,684).

B. GROUND OF REJECTION 2 (Claims 2-4, 21-23 and 40-42)

The Final Office Action rejects claims 2-4, 21-23 and 40-42 under 35 U.S.C. 103(a) as being allegedly unpatentable over *Spofford* et al. (U.S. Patent 5,913,037) in view of *Dobbins* et al. (U.S. Patent 5,951,649) in view of *Pearson* (U.S. Patent 6,023,684), as applied to claim 1, and further in view of *Whitehead* et al. (U.S. Patent 6,085,030).

C. GROUND OF REJECTION 3 (Claims 5-8, 10-18, 24-37, and 43-56)

The Final Office Action rejects claims 5-8, 10-18, 24-37, and 43-56 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Spofford* et al. (U.S. Patent 5,913,037) in view of *Dobbins* et al. (U.S. Patent 5,951,649) in view of *Pearson* (U.S. Patent 6,023,684), as applied to claim 1, and further in view of *Ferguson* et al. (U.S. Patent 6,016,499).

D. GROUND OF REJECTION 4 (Claims 19, 38, and 57)

The Final Office Action rejects claims 19, 38, and 57 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Spofford* et al. (U.S. Patent 5,913,037) in view of *Dobbins* et al. (U.S. Patent 5,951,649) in view of *Pearson* (U.S. Patent 6,023,684), as applied to claim 1, in view of *Ferguson* et al. (U.S. Patent 6,016,499), and in view of Admitted Prior Art.

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1, 9, 20, and 39)

The Final Office Action rejects claims 1, 9, 20 and 39 under 35 U.S.C. § 103(a) as being allegedly unpatentable over *Spofford et al.* (U.S. Patent 5,913,037), hereinafter referred to as *Spofford*, in view of *Dobbins et al.* (U.S. Patent 5,951,649), hereinafter referred to as *Dobbins* and further in view of *Pearson* (U.S. Patent 6,023,684). This rejection is respectfully traversed.

A.1. Claims 1, 9, 20, and 39

As to independent claims 1, 9, 20 and 39, the Final Office Action states:

> **As to claim 1,** *Spofford* teaches the invention substantially as claimed including: OID (OID, col 2, ln 59-67, col 6, ln 1-45, col 4, ln 1-9, col 7, ln 20-62, col 8, ln 15-52), abstraction layer (MIB manager, col 2, ln 59-67/ col 6, ln 1-45/ col 4, ln 1-9/ col 7, ln 20-62/ col 8, ln 15-52/ col 11, ln 1-30/ col 12, ln 40-67), an OID tree structure (col 2, ln 59-67/ col 6, ln 1-45, col 4, ln 1-9/ col 7, ln 20-62/ col 8, ln 15-52/ col 11, ln 1-30/ col 12, ln 40-67), query (query, col 11, ln 1-15), repository (the MIB 206, col 9, ln 40-41/ col 10, ln 58-59).
>
> *Spofford* does not explicitly teach the OID abstraction layer is capable of receiving queries for objects in two or more different protocols, registering the ODI tree structure with a registry associated with the OID. However, *Dobbins* teaches the OID abstraction layer is capable of receiving queries for objects in two or more different protocols (a standard interface for the Management Information Base for object access by any management protocol or other entity including SNMP, SNMPv2, DMP, col 16, ln 20-23), registering the ODI tree structure with a registry associated with the OID (Each specific managed object which is persistent is then created and calls the Persistent Object Manager to restore its values through the standard Managed Object base class ... will call the Persistent Object Manager 77 to store the value, col 20, ln 33-39/ all Base Resources are registered into one of these tables for management purposes, col 24, ln 49-53).
>
> It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine teaching of *Spofford* and *Dobbins* because *Dobbins*'s the OID abstraction layer is capable of receiving queries for objects in two or more different protocols, registering the ODI tree structure with a registry associated with the OID would improve the use of *Spofford* and *Dobbins*'s systems by providing a high availability of service, remoter management for supporting a number of different routing protocols.
>
> *Spofford* and *Dobbins* do not explicit teach mapping queries from multiple protocol interfaces to application programming interface (API) requests that the repository understands. However, *Pearson* teaches mapping queries from multiple protocol interfaces to application programming interface (API) requests that the repository understands (converted data from a parsed client request to a format compatible with the API for the

application service identified in the application service call, col 15-20/ converting client messages between the language supported by a client program and the language used to implement a application service, col 4, ln 67 to col 5, line 1-3/ convert s user queries from an Internet protocol to one compatible with a database ... the user queries to the appropriate query language format for the, col 2, ln 60-65/ presentation logic 80 communication with client program using HTML documents, other communication protocols may be used, col 11 ln 42-45/ client messages which are in the format of a known internet service, such as E-mail, Files transfer protocol, col 5, ln 60-65/ col 10, ln 32-37).

It would have been obvious to one of the ordinary skill in the art at the time the invention was made to combine the teaching of *Spofford*, *Dobbins* and *Pearson* because *Pearson*'s mapping queries from multiple protocol interfaces to application programming interface (API) requests that the repository understands would improve the efficiency of *Spofford* and *Dobbins*'s systems by allowing the customer with real time to access an execution of transaction commands over an open network without modifying a legacy database management system to support an increased number of users.

As to claim 9, it is an apparatus claim of claim 1; therefore it is rejected for the same reason at claim 1 above. In additional, *Pearson* teaches mapping queries from multiple protocol interfaces to application programming interface (API) requests that the repository understands (converted data from a parsed client request to a format compatible with the API for the application service identified in the application service call, col 15-20/ converting client messages between the language supported by a client program and the language used to implement a application service, col 4, ln 67 to col 5, line 1-3/ convert s user queries from an Internet protocol to one compatible with a database ... the user queries to the appropriate query language format for the, col 2, ln 60-65/ presentation logic 80 communication with client program using HTML documents, other communication protocols may be used, col 11 ln 42-45/ client messages which are in the format of a known internet service, such as E-mail, Files transfer protocol, col 5, ln 60-65/ col 10, ln 32-37), interpreting the first query according to the protocol recording to the protocol recognized by abstraction layer is one of the two or more different protocols (when a user wants to communicate an Internet service message such as e-mail, to a customer service representative, the message is provided through proxy firewall 54 to the e-mail service for delivery to a customer service computer 54. The customer service representative may be utilize information in the e-mail message to verify or correct user data through the application service 14, col 5, ln 61-65 and col 7, ln 27-35/ col 10, ln 32-39/ col 11, ln 15-20/ col 12, ln 58-60/ col 14, ln 35-43).

As to claims 20, 39, they are apparatus claims of claim 1; therefore, they are rejected for the same reason as claim 1 above.

Final Office Action dated June 13, 2005, pages 2-5.

Claim 1, which is representative of the other rejected independent claims 20 and 39 with regard to similarly recited subject matter, reads as follows:

1. A method on a server in a distributed data processing system for maintaining a logical composite repository of Object Identifier (OID) tree structures, the method comprising the steps of:

receiving, in an OID abstraction layer, an OID tree structure from a repository; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands;

registering the OID tree structure with a registry associated with the OID abstraction layer; and

adding the OID tree structure to a repository associated with the OID abstraction layer. (emphasis added)

Independent claim 9 is not an apparatus claim of claim 1; claim 9 is a method claim. Claim 9, which is representative of the other rejected independent claims 28 and 47 with regard to similarly recited subject matter, reads as follows:

9. A method on a server in a distributed data processing system for retrieving object data from a repository, comprising:

receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands;

interpreting the first query according to the protocol recognized by the OID abstraction layer, wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols;

locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and

retrieving the object data from the repository using an OID abstraction layer application program interface (API). (emphasis added)

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). For an invention to be prima facie obvious, the prior art must teach or suggest all claim limitations. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974).

Spofford, *Dobbins*, and *Pearson*, either taken individually or in combination, do not teach or suggest that "the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in independent claims 1, 20, and 39. Further, *Spofford*, *Dobbins*, and *Pearson*, either taken individually or in combination, do not teach or suggest

"receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in independent claim 9.

Spofford is directed to a dynamic management information base manager. A management information base (MIB) manager allows agents to add or delete objects to any level within the MIB tree by object identifier (OID). The MIB manager is a set of software interfaces, semantics, procedures, and data structures that work together to dynamically manage a tree of simple network management protocol (SNMP) data objects identified by an OID along with each object's value. SNMP is the only management protocol contemplated by *Spofford*. As stated in the Final Office Action, *Spofford* does not teach an OID abstraction layer that is capable of receiving queries for objects in two or more different protocols. The Final Office Action also states that *Spofford* does not teach mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands. Further, *Spofford* does not teach or suggest that "the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in claims 1, 20 and 39. Additionally, *Spofford* does not teach or suggest "receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in independent claim 9.

Dobbins is directed to a network interconnecting apparatus having a separate forwarding engine object at each interface. Each forwarding engine only knows the configuration information and how to receive and transmit packets on the one interface to which it corresponds. Each forwarding engine acts independently to process packets, yet each interacts together to collectively provide packet forwarding. An object-oriented architecture is provided that

distributes the critical function and system behavior into self-contained router objects. All router objects have the functions provided by the managed object class, which defines the methods and data for network management, built into the router object. The services and data normally external to the object are embedded or accessible within the object itself. The Final Office Action states that *Dobbins* does not teach mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands. *Dobbins* does not teach or suggest that "the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in claims 1, 20 and 39. Additionally, *Dobbins* does not teach or suggest "receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in independent claim 9.

In the rejection of independent claims 1, 9, 20 and 39, the Final Office Action refers to the following portion of *Dobbins*:

> A standard interface for the Management Information Base for object access by any management protocol or other entity including SNMP, SNMPv2, DMP, local device management, and other Managed Objects.

Dobbins, column 16, lines 20-23.

Although this portion of *Dobbins* states that any management protocol may be used to access an object using a standard interface for the Management Information Base (MIB), further analysis of *Dobbins* shows that *Dobbins* does not teach that multiple management protocols may be used to access an object using the standard interface for the MIB. Specifically, Figure 3A and Figure 3B of *Dobbins* reference an SNMP agent 228. Additionally, column 29, lines 31-33 of *Dobbins* states "SNMP operates by passing request to a device's internal database, the Management Information Base (MIB)." Thus, the preferred embodiment of *Dobbins* uses SNMP as the management protocol, but another management protocol may be used instead of SNMP. *Dobbins* does not teach or suggest an OID abstraction layer that is capable of receiving queries

for objects in two or more different protocols. Further, *Dobbins* does not teach or suggest an OID abstraction layer supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands, as recited in claims 1, 9, 20, and 39.

Additionally, *Dobbins* teaches that services and data normally external to the object are embedded or accessible within the object itself. Thus, there would not be a need for an OID abstraction layer to map queries from multiple protocol interfaces to application program interface (API) requests that the repository understands. Further, as stated in the Office Action dated December 14, 2004, *Spofford* and *Dobbins* do not teach API.

Pearson is directed to a three tier financial transaction system having a local data memory. The three tier system includes a client interface, an application service, a host interface, and a local data memory. The client interface communicates data messages between a client program and the financial transaction system. The client interface converts client requests to a format compatible with the application service so the application service may process client requests from client programs. At the initiation of a logical session with a client program, the application service refreshes data for the customer associated with the client program using data obtained from a back end processing system through the host interface. The data in the local data memory is then used by the application service for processing client requests during the logical session. Response data generated by the application service is provided to the client interface for presentation to the client program. Communication between the client program and the client interface is preferably performed over an open communication network. The local data memory permits the processing of the client service request to be decoupled from the updating of the back end processing system to improve response times for client request processing. See *Pearson*, abstract. *Pearson* does not teach or suggest that "the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in claims 1, 20 and 39. Additionally, *Pearson* does not teach or suggest "receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different

protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in independent claim 9.

In the rejection of independent claims 1, 9, 20 and 39, the Final Office Action refers to the following portions of *Pearson*:

> A gateway can be an application program or a separate system which converts user queries from an Internet protocol to one compatible with a database coupled to the gateway. If more than one database is coupled to the gateway, the gateway performs the function of converting the user queries to the appropriate query language format for the database coupled to the gateway.

Pearson, column 2, lines 60-65.

> The client interface services also include personality libraries for converting client messages between the language supported by a client program and the language used to implement an application service. For example, a client program may provide client messages or requests in JAVA, Active X, or other language commonly encountered on the Internet.

Pearson, column 4, line 67, though column 5, line 3.

> Client messages which are in the format of a known internet service, such as E-mail, file transfer protocol (FTP), or Telnet messages, are delivered to a proxy firewall before being delivered to the server which supports the Internet service.

Pearson, column 5, lines 60-65.

> Firewall 54 permits customer service computers 52 which are coupled together through a computer network to utilize internet services, such as e-mail, World Wide Web, FTP, Telnet, Rlogin and Usenet in a secure manner. The system includes a network access controller that interrogates a connection request for a protected service to determine whether the request should be granted.

Pearson, column 10, lines 32-37.

> Personality library 82 converts data from a parsed client request to a format compatible with the API for the application service identified in the application service call. For example, client interface 12 may receive a client request in an HTML file from a client program 30.

Pearson, column 11, lines 15-20.

> Continuing the example, personality library 82 of client interface 12 then converts the response data to a form compatible for HTML files and presentation logic 80 builds an HTML document that is sent to client program 30. Although the preferred presentation logic communicates with client programs using HTML documents, other communication protocols may be used.

Pearson, column 11, lines 42-45.

These portions of *Pearson* do not teach or suggest that "the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in claims 1, 20 and 39. Additionally, *Pearson* does not teach or suggest "receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in independent claim 9. Further, *Pearson* does not even mention protocols, such as SNMP, LDAP, and CIM/XML, for maintaining a logical composite repository of OID tree structures or retrieving object data from a repository.

Spofford, *Dobbins*, and *Pearson* fail to teach or suggest that "the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in claims 1, 20, and 39. With respect to claim 9, *Spofford*, *Dobbins*, and *Pearson* fail to teach or suggest "receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands." Therefore, the alleged combination of *Spofford*, *Dobbins*, and *Pearson* does not teach or suggest these features, as recited in independent claims 1, 9, 20, and 39.

Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally

available to one of ordinary skill in the art. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988); *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992).

One of ordinary skill in the art would not combine *Spofford*, *Dobbins*, and *Pearson* when the references are considered as a whole. In considering the references as a whole, one of ordinary skill in the art would take into account the problems recognized and solved. The present invention recognizes the problem of processing queries referencing an object using two or more different protocols. *Spofford* is directed toward modifying a system without resetting the system or powering it down, while dynamically adding to or modifying the management structure of the system to enable management of new or modified devices and resources (see *Spofford*, column 2, lines 49-56). *Dobbins* is directed to an object-oriented system which utilizes common protocol-independent base objects to instantiate protocol-specific objects and distributes the critical function and system behavior into autonomous objects (see *Dobbins*, column 1, lines 14-20). *Pearson* is directed to a three tier financial transaction system for interfacing client programs over an open network to legacy databases in financial transaction computer systems (see *Pearson*, column 1, lines 5-10). One of ordinary skill in the art would therefore not be motivated to combine or modify the references in the manner required to form the solution disclosed in the present invention.

Furthermore, as noted above, there is no teaching or suggestion in the references as to the desirability of including the features from the other references. There is no motivation cited in *Spofford* to include the object-oriented system of *Dobbins* which utilizes common protocol-independent base objects to instantiate protocol-specific objects and the three tier financial transaction system of *Pearson*. The Examiner alleges that the motivation would be to improve the efficiency of *Spofford*'s and *Dobbins*'s systems by allowing the customer with real time to access an execution of transaction commands over an open network without modifying a legacy database management system to support an increased number of users. Appellant respectfully disagrees that this would be a proper motivation for combining *Spofford*, *Dobbins*, and *Pearson*. As the Examiner has failed to demonstrate any motivation or incentive in the prior art to combine and modify the references so as to achieve the claimed invention, the alleged combination can only be the result of impermissible hindsight reconstruction using Appellant's own disclosure as a guide. While Appellant understands that all examination entails some measure of hindsight, when the rejection is based completely on hindsight, as in the present case,

to the exclusion of what can be gleaned from the references, then the rejection is improper and should be withdrawn.

Thus, *Spofford*, *Dobbins*, and *Pearson*, either taken individually or in combination, do not teach or suggest the features of claims 1, 9, 20, and 39. Accordingly, Appellant respectfully requests withdrawal of the rejection of claims 1, 9, 20, and 39 under 35 U.S.C. § 103(a).

B. GROUND OF REJECTION 2 (Claims 2-4, 21-23, and 40-42)

The Final Office Action rejects claims 2-4, 21-23 and 40-42 under 35 U.S.C. 103(a) as being allegedly unpatentable over *Spofford* in view of *Dobbins* in view of *Pearson*, as applied to claim 1, and further in view of *Whitehead et al.* (U.S. Patent 6,085,030), hereinafter referred to as *Whitehead*.

B.1. Claims 2-4, 21-23, and 40-42

Since claims 2-4, 21-23 and 40-42 depend from independent claims 1, 20 and 39, respectively, the same distinctions between *Spofford*, *Dobbins*, *Pearson*, and the invention recited in claims 1, 20 and 39, apply to dependent claims 2-4, 21-23 and 40-42. In addition, *Whitehead* does not provide for the deficiencies of *Spofford*, *Dobbins*, and *Pearson* with regard to independent claims 1, 20 and 39. *Whitehead* is directed toward a network component server that provides an object-neutral global component registry. *Whitehead* is cited for teaching an anchor point. *Whitehead* does not teach or suggest an OID abstraction layer that is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands. Thus, any alleged combination of *Whitehead* with *Spofford*, *Dobbins*, and *Pearson* still would not result in the invention recited in claims 1, 20 and 39 from which claims 2-4, 21-23 and 40-42 depend. Accordingly, Appellant respectfully requests withdrawal of the rejection of claims 2-4, 21-23 and 40-42 under 35 U.S.C. § 103(a).

C. GROUND OF REJECTION 3 (Claims 5-8, 10-18, 24-37, and 43-56)

The Final Office Action rejects claims 5-8, 10-18, 24-37, and 43-56 under 35 U.S.C. 103(a) as being allegedly unpatentable over *Spofford* in view of *Dobbins* in view of *Pearson*, as applied to claim 1, and further in view of *Ferguson et al.* (U.S. Patent 6,016,499), hereinafter referred to as *Ferguson*. This rejection is respectfully traversed.

C.1. Claims 5-8, 10-18, 24-37, and 43-56

Independent claim 9 is not an apparatus claim of claim 1; claim 9 is a method claim, which is representative of the other rejected independent claims 28 and 47 with regard to similarly recited subject matter. As to independent claims 9, 28 and 47, the Final Office Action states:

> **As to claim 9,** it is an apparatus claim of claim 1; therefore it is rejected for the same reason at claim 1 above. In additional, *Pearson* teaches mapping queries from multiple protocol interfaces to application programming interface (API) requests that the repository understands (converted data from a parsed client request to a format compatible with the API for the application service identified in the application service call, col 15-20/ converting client messages between the language supported by a client program and the language used to implement a application service, col 4, ln 67 to col 5, line 1-3/ convert s user queries from an Internet protocol to one compatible with a database ... the user queries to the appropriate query language format for the, col 2, ln 60-65/ presentation logic 80 communication with client program using HTML documents, other communication protocols may be used, col 11 ln 42-45/ client messages which are in the format of a known internet service, such as E-mail, Files transfer protocol, col 5, ln 60-65/ col 10, ln 32-37), interpreting the first query according to the protocol recording to the protocol recognized by abstraction layer is one of the two or more different protocols (when a user wants to communicate an Internet service message such as e-mail, to a customer service representative, the message is provided through proxy firewall 54 to the e-mail service for delivery to a customer service computer 54. The customer service representative may be utilize information in the e-mail message to verify or correct user data through the application service 14, col 5, ln 61-65 and col 7, ln 27-35/ col 10, ln 32-39/ col 11, ln 15-20/ col 12, ln 58-60/ col 14, ln 35-43). ...
>
> **As to claims 24-37, 43-56,** they are apparatus claims of claims 5-9, 10-18; therefore, they are rejected for the same reason as claims 5-9, 10-18 above.

Final Office Action dated June 13, 2005, pages 4-5 and 8.

The previous Office Action, dated December 14, 2004, states the following with regard to independent claim 9:

> **As to claim 9,** *Spofford* teaches a first query (a query, col 10, ln 25-67 to col 11, ln

1-16), the object data (the objects, col 10, ln 25-67 to col 11, ln 1-16), a request (request, col 10, ln 25-67), a protocol (SNMP, col 1, ln 1-35/ protocol, col 5, ln 5-67/ col 6, ln 1-67), OID (OID, col 2, ln 59-67, col 6, ln 1-45, col 4, ln 1-9, col 7, ln 20-62, col 8, ln 15-52), abstraction layer (MIB manager, col 2, ln 59-67/ col 6, ln 1-45/col 4, ln 1-9/ col 7, ln 20-62/ col 8, ln 15-52/ col 11, ln 1-30/ col 12, ln 40-67).

Spofford does not explicitly teach the OID abstraction layer is capable of receiving queries for objects in two or more different protocols, locating a repository that contain the object data requested in the first query based on a registry. However, *Dobbins* teaches the OID abstraction layer is capable of receiving queries for objects in two or more different protocols (a standard interface for the Management Information Base for object access by any management protocol or other entity including SNMP, SNMPv2, DMP, col 16, ln 20-23), locating a repository that contain the object with that database, by using the object's identifier (OID) ... the format of these requests by proving a textual representation to these OID's, which are easier for the user to digest, col 29, ln 34-40 and col 42-45).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine teaching of *Spofford* and *Dobbins* because *Dobbins*'s the OID abstraction layer is capable of receiving queries for objects in two or more different protocols, registering the ODI tree structure with a registry associated with the OID would provide a high availability of service, remoter management for supporting a number of different routing protocols.

Office Action dated December 14, 2004, pages 5-6.

Claim 9, which is representative of the other rejected independent claims 28 and 47 with regard to similarly recited subject matter, reads as follows:

9. A method on a server in a distributed data processing system for retrieving object data from a repository, comprising:

receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands;

interpreting the first query according to the protocol recognized by the OID abstraction layer, wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols;

locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and

retrieving the object data from the repository using an OID abstraction layer application program interface (API). (emphasis added)

As discussed above, *Spofford*, *Dobbins*, and *Pearson*, taken individually or in combination, fail to teach or suggest that "the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different

protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in independent claims 1, 9, 20, 28, 39 and 47. Similarly, *Spofford*, *Dobbins*, and *Pearson*, either taken individually or in combination, do not teach or suggest "interpreting the first query according to the protocol recognized by the OID abstraction layer, wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols," as recited in claims 9, 28 and 47. In addition, *Ferguson* does not provide for the deficiencies of *Spofford*, *Dobbins*, and *Pearson* with regard to independent claims 1, 9, 20, 28, 39 and 47.

Ferguson is directed to a system and method for accessing a directory services repository. *Ferguson* does generally teach application program interfaces (API) and, more specifically, teaches an API that includes at least one callable element that is capable of accessing a component of a repository in response to being called and a driver that is capable of translating a database language statement, such as an SQL statement, into an executable API sequence. However, *Ferguson* does not teach or suggest that "the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in claims 1, 9, 20, 28, 39 and 47. Further, *Ferguson* does not teach or suggest "interpreting the first query according to the protocol recognized by the OID abstraction layer, wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols," as recited in independent claims 9, 28 and 47.

Spofford, *Dobbins*, *Pearson*, and *Ferguson* do not teach or suggest that "the OID abstraction layer that is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in independent claims 1, 9, 20, 28, 39 and 47. Therefore, the alleged combination of *Spofford*, *Dobbins*, *Pearson*, and *Ferguson* does not teach or suggest these features, as recited in independent claims 1, 9, 20, 28, 39 and 47.

Additionally, *Spofford*, *Dobbins*, *Pearson*, and *Ferguson* do not teach or suggest "interpreting the first query according to the protocol recognized by the OID abstraction layer, wherein the protocol recognized by the OID abstraction layer is one of the two or more different

protocols," as recited in claims 9, 28, and 47. Therefore, the alleged combination of *Spofford*, *Dobbins*, *Pearson*, and *Ferguson* does not teach or suggest these features, as recited in claims 9, 28 and 47.

Thus, *Spofford*, *Dobbins*, *Pearson*, and *Ferguson*, taken individually or in combination, do not teach or suggest that "the OID abstraction layer that is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands," as recited in independent claims 1, 9, 20, 28, 39 and 47 and do not teach or suggest "interpreting the first query according to the protocol recognized by the OID abstraction layer, wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols," as recited in independent claims 9, 28 and 47. Therefore, *Spofford*, *Dobbins*, *Pearson*, and *Ferguson*, taken individually or in combination, do not teach or suggest the features of dependent claims 5-8, 10-18, 24-27, 29-37, 43-46 and 48-56 at least by virtue of their dependency on claims 1, 9, 20, 28, 39 and 47, respectively. Accordingly, Appellant respectfully requests withdrawal of the rejection of claims 5-8, 10-18, 24-37, and 43-56 under 35 U.S.C. § 103(a).

C.2. Claims 6, 25, and 44

In addition, Appellant respectfully submits that claims 6, 25, and 44 are independently distinguishable from the *Spofford*, *Dobbins*, *Pearson*, and *Ferguson* references. Claim 6 depends from claim 1; claim 25 depends from claim 20; and claim 44 depends from claim 39. *Spofford*, *Dobbins*, *Pearson*, and *Ferguson*, either taken alone or in combination, do not teach or suggest that the OID abstraction layer receives the information retrieved from the repository through the API and encapsulates the information in a reply message to a target protocol interface, wherein the reply message is formatted for an appropriate protocol for the target protocol interface, and wherein the appropriate protocol is one of the two or more different protocols. The cited portions of *Spofford* and *Ferguson* only teach a protocol interface, a request, a reply message and an API rather than teaching that the reply message is formatted for an appropriate protocol for the protocol interface, and wherein the appropriate protocol is one of the two or more different protocols, as recited in claims 6, 25 and 44.

C.3. Claims 7-8, 26-27 and 45-46

Additionally, Appellant respectfully submits that claims 7-8, 26-27 and 45-46 are independently distinguishable from the *Spofford*, *Dobbins*, *Pearson*, and *Ferguson* references. Claims 7-8 depend from claim 1; claims 26-27 depend from claim 20; and claims 45-46 depend from claim 39. With regard to claims 7-8, 26-27 and 45-46, the cited portion of *Ferguson* only teaches translating the API result into a relational database result. *Spofford*, *Dobbins*, *Pearson*, and *Ferguson*, either taken alone or in combination, do not teach or suggest that the OID abstraction layer receives a request for object data from a requesting protocol interface, interprets the request according to a protocol of the requesting protocol interface, wherein the protocol is one of the two or more different protocols, converts the request into an application program interface (API) request that is forwarded to the repository, and receives an API reply from the repository having the object data, as recited in claims 7, 26 and 45. The applied references also fail to teach or fairly suggest that the OID abstraction layer reformats the object data in a reply message according to the protocol and sends the reply message to the protocol interface, as recited in claims 8, 27 and 46.

C.4. Claims 14, 33 and 52

In addition to the above, Appellant respectfully submits that claims 14, 33, and 52 are independently distinguishable from the *Spofford*, *Dobbins*, *Pearson*, and *Ferguson* references. Claim 14 depends from claim 9; claim 33 depends from claim 28; and claim 52 depends from claim 47. *Spofford*, *Dobbins*, *Pearson*, and *Ferguson*, either taken alone or in combination, do not teach or suggest that the first reply is transformed into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer, and wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols.

C.5. Claims 16, 35 and 54

In addition, Appellant respectfully submits that claims 16, 35, and 54 are independently

distinguishable from the *Spofford*, *Dobbins*, *Pearson*, and *Ferguson* references. Claim 16 depends from claim 9; claim 35 depends from claim 28; and claim 54 depends from claim 47. *Spofford*, *Dobbins*, *Pearson*, and *Ferguson*, either taken alone or in combination, do not teach or suggest that each repository in the plurality of repositories contains information representing an Object Identifier (OID) subtree structure, and wherein the plurality of repositories are formatted to support the two or more different protocols.

D. GROUND OF REJECTION 4 (Claims 19, 38, and 57)

The Final Office Action rejects claims 19, 38, and 57 are rejected under 35 U.S.C. 103(a) as being allegedly unpatentable over *Spofford* in view of *Dobbins* in view of *Pearson*, as applied to claim 1, in view of *Ferguson*, and in view of *Admitted Prior Art*. This rejection is respectfully traversed.

D.1. Claims 19, 38, and 57

Since claims 19, 38 and 57 depend from independent claims 9, 28 and 47, respectively, the same distinctions between *Spofford*, *Dobbins*, *Pearson*, *Ferguson* and the invention recited in claims 9, 28 and 47, apply to dependent claims 19, 38 and 57. Accordingly, Appellant respectfully requests withdrawal of the rejection of claims 19, 38 and 57 under 35 U.S.C. § 103(a).

Further, there is no suggestion or motivation whatsoever in *Spofford*, *Dobbins*, *Pearson*, and *Ferguson* for using CIM/XML. The Final Office Action states that *Spofford*, *Dobbins*, *Pearson*, and *Ferguson* do not teach CIM/XML. The mere fact that a prior art reference can be readily modified does not make the modification obvious unless the prior art suggested the desirability of the modification. *In re Laskowski*, 871 F.2d 115, 10 U.S.P.Q.2d 1397 (Fed. Cir. 1989) and also *see In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992) and *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1993). The Final Office Action may not merely state that the modification would have been obvious to one of ordinary skill in the art without pointing out in the prior art a suggestion of the desirability of the proposed modification. In this case, the only suggestion or motivation for making the proposed modification is found in

Appellant's own specification. As a result, absent any teaching, suggestion, or incentive from the prior art to make the proposed modification, the presently claimed invention can be reached only through the an impermissible use of hindsight with the benefit of Appellant's disclosure a model for the needed changes.

Gerald H. Glanzman
Reg. No. 25,035
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

GHG/vja

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A method on a server in a distributed data processing system for maintaining a logical composite repository of Object Identifier (OID) tree structures, the method comprising the steps of:

receiving, in an OID abstraction layer, an OID tree structure from a repository; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands;

registering the OID tree structure with a registry associated with the OID abstraction layer; and

adding the OID tree structure to a repository associated with the OID abstraction layer.

2. The method of claim 1, wherein the registry associated with the OID abstraction layer provides information identifying an anchor point in the OID subtree structure to be maintained by the repository.

3. The method of claim 2, wherein if the anchor point of the OID subtree structure is already registered with the OID abstraction layer, the registry is overwritten.

4. The method of claim 2, wherein if a query is received for an object that has an Object Identifier that is below a registered anchor point in an OID tree structure, the OID abstraction layer identifies a repository that maintains object information for the requested object based on the registered anchor point.

5. The method of claim 1, wherein the repository is configured such that the repository recognizes requests from an application program interface (API) associated with the OID abstraction layer and sends reply messages to the API containing information retrieved from the repository.

6. The method of claim 5, wherein the OID abstraction layer receives the information retrieved from the repository through the API and encapsulates the information in a reply message to a target protocol interface, wherein the reply message is formatted for an appropriate protocol for the target protocol interface, and wherein the appropriate protocol is one of the two or more different protocols.

7. The method of claim 1, wherein the OID abstraction layer receives a request for object data from a requesting protocol interface, interprets the request according to a protocol of the requesting protocol interface, wherein the protocol of the requesting protocol interface is one of the two or more different protocols, converts the request into an application program interface (API) request that is forwarded to the repository, and receives an API reply from the repository having the object data.

8. The method of claim 7, wherein the OID abstraction layer reformats the object data in a reply message according to the protocol of the requesting protocol interface and sends the reply message to the requesting protocol interface.

9. A method on a server in a distributed data processing system for retrieving object data from a repository, comprising:

receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands;

interpreting the first query according to the protocol recognized by the OID abstraction layer, wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols;

locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and

retrieving the object data from the repository using an OID abstraction layer application program interface (API).

10. The method of claim 9, wherein the first query is mapped into a second query, wherein the second query is consistent with an application program interface (API) associated with the OID abstraction layer.

11. The method of claim 10, wherein if the first query cannot be mapped into a second query due to a limitation of the repository that contains the object associated with the first query, then the first query cannot be satisfied.

12. The method of claim 10, wherein the second query is sent to the repository that contains the object associated with the first query.

13. The method of claim 12, wherein a first reply is received at the API associated with the OID abstraction layer from the repository that contains the object associated with the first query.

14. The method of claim 13, wherein the first reply is transformed into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer, and wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols.

15. The method of claim 14, wherein the second reply is sent to the requester in the distributed data processing system.

16. The method of claim 9, wherein each repository in a plurality of repositories contains information representing an Object Identifier (OID) subtree structure, and wherein the plurality of repositories are formatted to support the two or more different protocols.

17. The method of claim 9, wherein Simple Network Management Protocol (SNMP) is a protocol recognized by the OID abstraction layer.

18. The method of claim 9, wherein Lightweight Directory Access Protocol (LDAP) is a protocol recognized by the OID abstraction layer.

19. The method of claim 9, wherein Common Information Model used in conjunction with eXtendable Markup Language (CIM/XML) is a protocol recognized by the OID abstraction layer.

20. An apparatus on a server in a distributed data processing system for maintaining a logical composite repository of Object Identifier (OID) tree structures, the apparatus comprising:

an OID abstraction layer that receives an OID tree structure from a repository; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands;

a registry, associated with the OID abstraction layer, that registers the OID tree structure; and

an adding means for adding the OID tree structure to a repository associated with the OID abstraction layer.

21. The apparatus of claim 20, wherein the registry provides information identifying an anchor point in the OID tree structure to be maintained by the repository.

22. The apparatus of claim 21, wherein if the anchor point of the OID tree structure is already registered in the registry, then the registry overwrites the previous entry.

23. The apparatus of claim 21, wherein, if the OID abstraction layer receives a query for an object that has an Object Identifier that is below a registered anchor point in an OID tree structure, the registry in the OID abstraction layer identifies a repository that maintains object information for the requested object based on the registered anchor point.

24. The apparatus of claim 20, wherein the repository is configured such that the repository recognizes requests received from an application program interface (API) associated with the OID abstraction layer and sends reply messages to the API containing information retrieved from the repository.

25. The apparatus of claim 24, wherein the OID abstraction layer receives the information retrieved from the repositories through the API and encapsulates the information in a reply message to a target protocol interface, wherein the reply message is formatted for an appropriate protocol for the target protocol interface, and wherein the appropriate protocol is one of the two or more different protocols.

26. The apparatus of claim 20, wherein the OID abstraction layer receives a request for object data from a requesting protocol interface, interprets the request according to a protocol of the requesting protocol interface, wherein the protocol of the requesting protocol interface is one of

the two or more different protocols, converts the request into an application program interface (API) request that is forwarded to the repository, and receives an API reply from the repository having the object data.

27. The apparatus of claim 26, wherein the OID abstraction layer encapsulates the object data in a reply message according to the protocol of the requesting protocol interface and sends the reply message to the requesting protocol interface.

28. An apparatus on a server in a distributed data processing system for retrieving object data from a repository, comprising:

a receiving means for receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands;

a interpreting means for interpreting the first query according to the protocol recognized by the OID abstraction layer, wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols;

a locating means for locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and

a retrieving means for retrieving the object data from the repository using an OID abstraction layer application program interface (API).

29. The apparatus of claim 28, further comprising a mapping means for mapping the first query into a second query, wherein the second query is consistent with an application program interface (API) associated with the OID abstraction layer.

30. The apparatus of claim 29, wherein if the mapping means cannot map the first query into a second query due to a limitation of the repository that contains the object associated with the first query, then the first query cannot be satisfied.

31. The apparatus of claim 29, further comprising a first sending means, in the OID abstraction layer, that sends the second query to a repository that contains the object associated with the first query.

32. The apparatus of claim 31, wherein the retrieving means receives a first reply at the API from the repository that contains the object associated with the first query.

33. The apparatus of claim 32, further comprising a transforming means, in the OID abstraction layer, that transforms the first reply into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer, and wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols.

34. The apparatus of claim 33, further comprising a second sending means, in the OID abstraction layer, that sends the second reply to the requester in the distributed data processing system.

35. The apparatus of claim 28, wherein each repository in a plurality of repositories contains Object Identifier (OID) tree structures, and wherein the plurality of repositories are formatted to support the two or more different protocols.

36. The apparatus of claim 28, wherein the receiving means recognizes a Simple Network Management Protocol (SNMP) query.

37. The apparatus of claim 28, wherein the receiving means recognizes a Lightweight Directory Access Protocol (LDAP) query.

38. The apparatus of claim 28, wherein the receiving means recognizes a Common Information Model used in conjunction with eXtendable Markup Language (CIM/XML) query.

39. (Currently Amended): A computer program product in a computer readable medium for maintaining a repository of Object Identifier (OID) tree structures, comprising:

instructions for receiving, in an OID abstraction layer, an OID tree structure from a repository; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries

from multiple protocol interfaces to application program interface (API) requests that the repository understands;

instructions for registering the OID tree structure with a registry associated with the OID abstraction layer; and

instructions for adding the OID tree structure to a repository associated with the OID abstraction layer.

40. The computer program product of claim 39, further comprising instructions for maintaining the registry associated with the OID abstraction layer and providing information identifying an anchor point in the OID tree structure to be maintained by the repository.

41. The computer program product of claim 40, wherein if the anchor point of the OID tree structure is already registered with the OID abstraction layer, the instructions for registering overwrites the previous entry.

42. The computer program product of claim 40, further comprising instructions for identifying a repository that maintains object information for the requested object based on the registered anchor point if a query is received for an object that has an Object Identifier that is below a registered anchor point in an OID tree structure.

43. The computer program product of claim 39, further comprising instructions for configuring the repository to recognize requests from an application program interface (API)

associated with the OID abstraction layer and to send reply messages to the API containing information retrieved from the repository.

44. The computer program product of claim 43, further comprising instructions for receiving the information retrieved from the repository, through the API, and encapsulating the information in a reply message to a target protocol interface, wherein the reply message is formatted for an appropriate protocol for the target protocol interface, and wherein the appropriate protocol is one of the two or more different protocols.

45. The computer program product of claim 39, further comprising:

instructions for receiving a request for object data from a requesting protocol interface;

instructions for interpreting the request according to a protocol of the requesting protocol interface, wherein the protocol of the requesting protocol interface is one of the two or more different protocols,

instructions for converting the request into an application program interface (API) request which is forwarded to the subtree repository; and

instructions for receiving an API reply from the subtree repository having the object data.

46. The computer program product of claim 45, further comprising instructions for encapsulating the object data in a reply message according to the protocol of the requesting protocol interface and sending the reply message to the requesting protocol interface.

47. A computer program product in a computer readable medium for retrieving object data from a repository, comprising:

instructions for receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols and supports the two or more different protocols by mapping queries from multiple protocol interfaces to application program interface (API) requests that the repository understands;

instructions for interpreting the first query according to the protocol recognized by the OID abstraction layer, wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols;

instructions for locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and

instructions for retrieving the object data from the repository using an OID abstraction layer application program interface (API).

48. The computer program product of claim 47, wherein the instructions for receiving the first query map the first query into a second query, wherein the second query is consistent with an application program interface (API) associated with the OID abstraction layer.

49. The computer program product of claim 48, wherein if the instructions for receiving the first query map cannot map the first query into a second query due to a limitation of the

repository that contains the object associated with the first query, then the first query cannot be satisfied.

50. The computer program product of claim 48, further comprising instructions for sending the second query to the repository that contains the object associated with the first query.

51. The computer program product of claim 50, further comprising instructions for receiving a first reply at the API associated with the OID abstraction layer from the repository that contains the object associated with the first query.

52. The computer program product of claim 51, further comprising instructions for transforming the first reply into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer, and wherein the protocol recognized by the OID abstraction layer is one of the two or more different protocols.

53. The computer program product of claim 52, further comprising instructions for sending the second reply to the requester in the distributed data processing system.

54. The computer program product of claim 47, wherein each repository in a plurality repositories contains Object Identifier (OID) tree structures, and wherein the plurality of repositories are formatted to support the two or more different protocols.

55. The computer program product of claim 47, wherein instructions for receiving a first query recognize a Simple Network Management Protocol (SNMP) query.

56. The computer program product of claim 47, wherein instructions for receiving a first query recognize a Lightweight Directory Access Protocol (LDAP) query.

57. The computer program product of claim 47, wherein instructions for receiving a first query recognize a Common Information Model used in conjunction with eXtendable Markup Language (CIM/XML) query.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.